In the Begining . . .

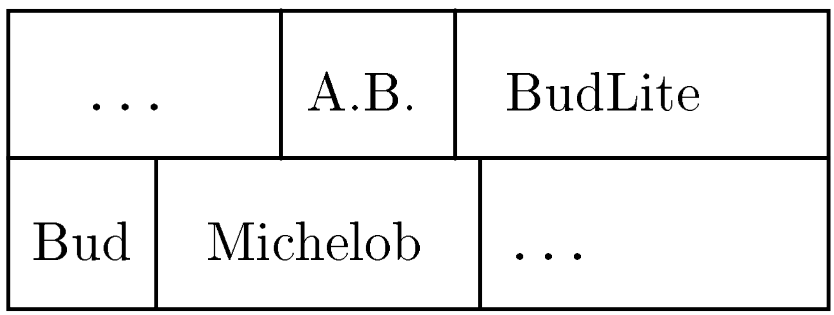
(Historical data models: Network and Hierarchical)

- The mid 1960's saw the first systems that used secondary storage for *querying* = retrieval by value, not by file name.
- Big difference: *secondary storage model of data* = data in blocks/pages, and major cost is retrieving or storing a *block*.
- *Unsolved problem*: Storing many-many relationships so they can be traversed efficiently in both directions.
  - Easy in RAM model: linked lists of successors, predecessors, e.g.
  - Many-one is easy in secondary-storage model.

Example

Many-one relationship from beers to manufacturers.

- Store each beer following its manufacturer, so "find the Anheuser-Busch beers" can be answered by retrieving them all on one or a few blocks.



Network Model

Essentially entity sets and binary, many-one relationships.

- Replace a many-many relationship by a connecting E.S. and two many-one relationships.
- Ditto for *any* 3-way (or more) relationship.

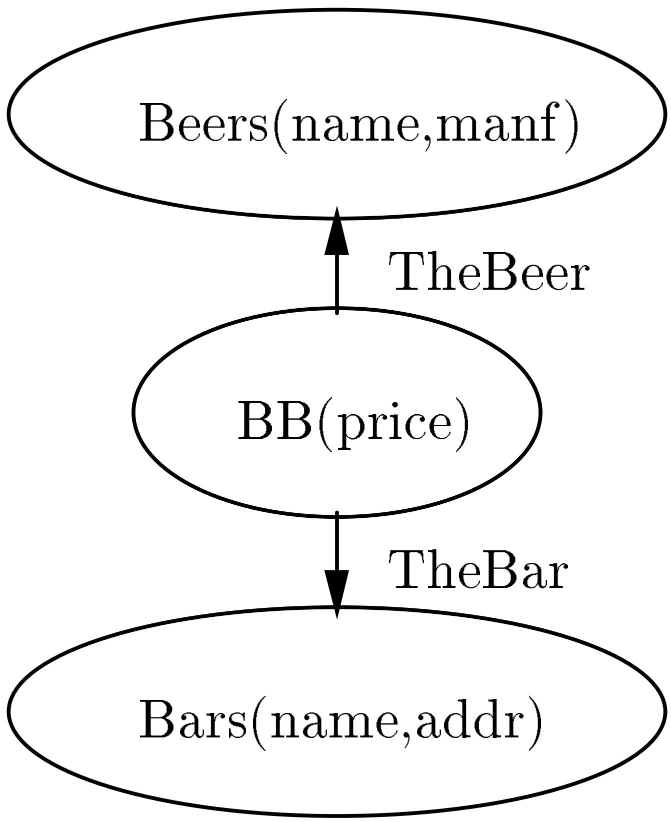
Terminology

- Entity set → *Logical Record Type* (LRT).
- Many-one relationship → *Link.*
  - ❖ Terminology useful to this day: *owner* = one, *member* = many, e.g., a manufacturer record "owns" beer records.

Notation

We show each LRT as an oval, with its name and any associated attributes in parentheses.

Example

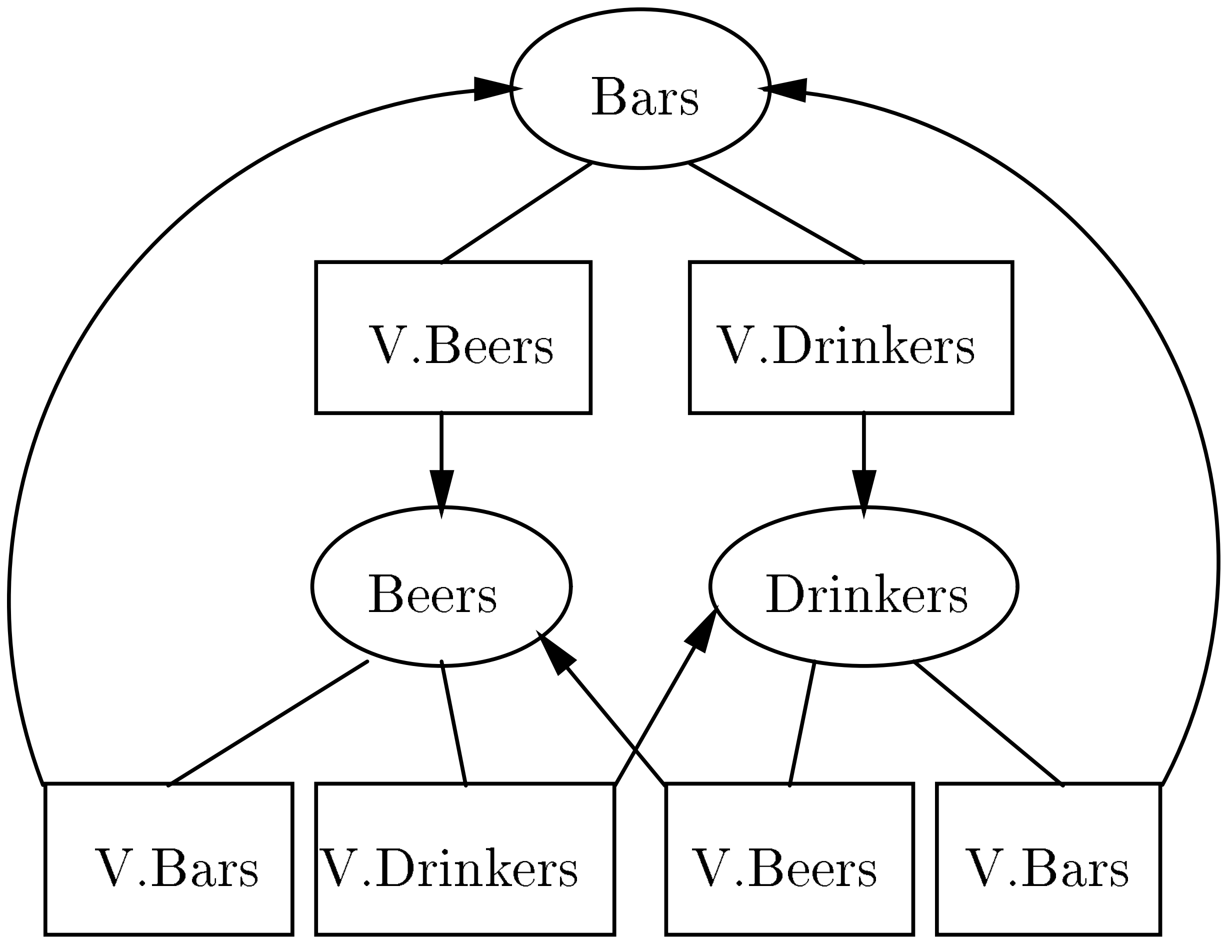


Hierarchical Model

Used in major early DBMS's, including IBM's IMS, which is still supported today.

- Network model, restricted to a forest, where owners are parents of children.
- Adds *Virtual LRT* to handle many-many relationships.
    - Think of V. LRT as representing pointers.

Example

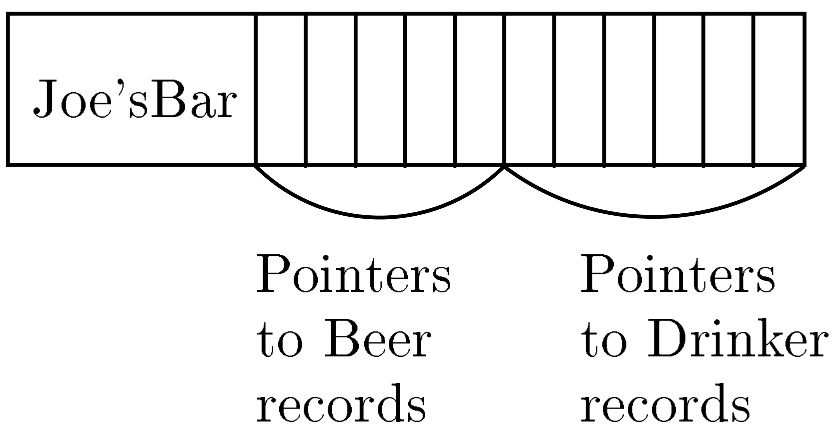


- Note: parenthesized list of attributes (for real LRT only) is omitted.

Intended Storage Structure

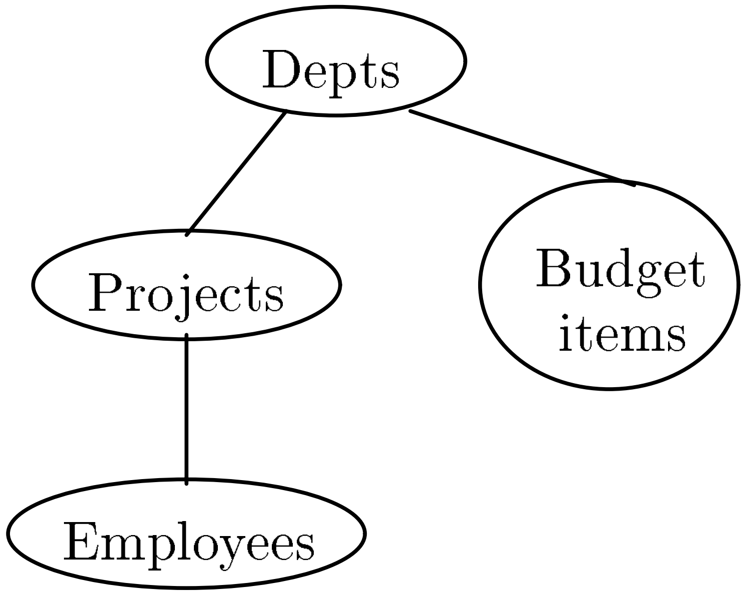
- Each LRT (not virtual) has its own attributes followed by representations for all its children.
  - ✥ The child LRT's may in turn have nested child-records, etc.
  - ✥ A virtual-LRT-child is represented by a pointer only.

Example: Typical Bar Record

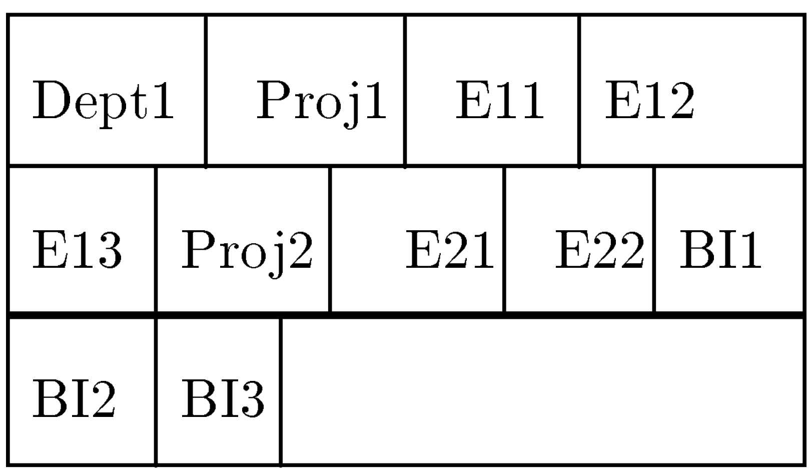


- No help in secondary storage model when going from a bar to either its beers or its drinkers.

Example Where Hierarchical Model Wins



- Typical stored records make efficient queries that go Dept → Budget Item or Project → Employees.



Relational Model

- Table = relation.
- Column headers = *attributes*.
- Row = *tuple*

name	manf
WinterBrew	Pete's
BudLite	A.B.
...	...

`Beers`

- *Relation schema* = name(attributes) + other structure info., e.g., keys, other constraints. Example: `Beers(name, manf)`.
    - ❖ Order of attributes is arbitrary, but in practice we need to assume the order given in the relation schema.
- *Relation instance* is current set of rows for a relation schema.
- *Database schema* = collection of relation schemas.

Keys in Relations

An attribute or set of attributes K is a *key* for a relation R if we expect that in no instance of R will two different tuples agree on all the attributes of K.

- Indicate a key by underlining the key attributes.
- Example: If `name` is a key for `Beers`:

  Beers(<u>name</u>, manf)

Why Relations?

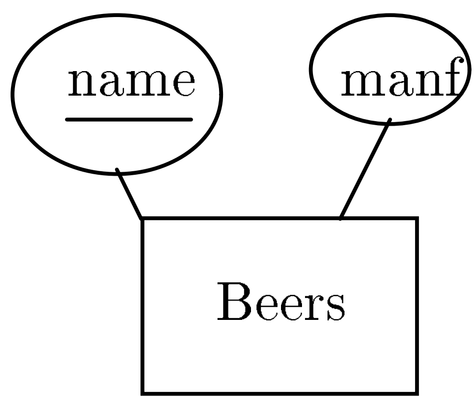
- Very simple model.
- *Often* a good match for the way we think about our data.
- Abstract model that underlies SQL, the most important language in DBMS's today.
    - But SQL uses "bags," while the abstract relational model is set-oriented.

Relational Design

Simplest approach (not always best): convert each E.S. to a relation and each relationship to a relation.

Entity Set → Relation

E.S. attributes become relational attributes.



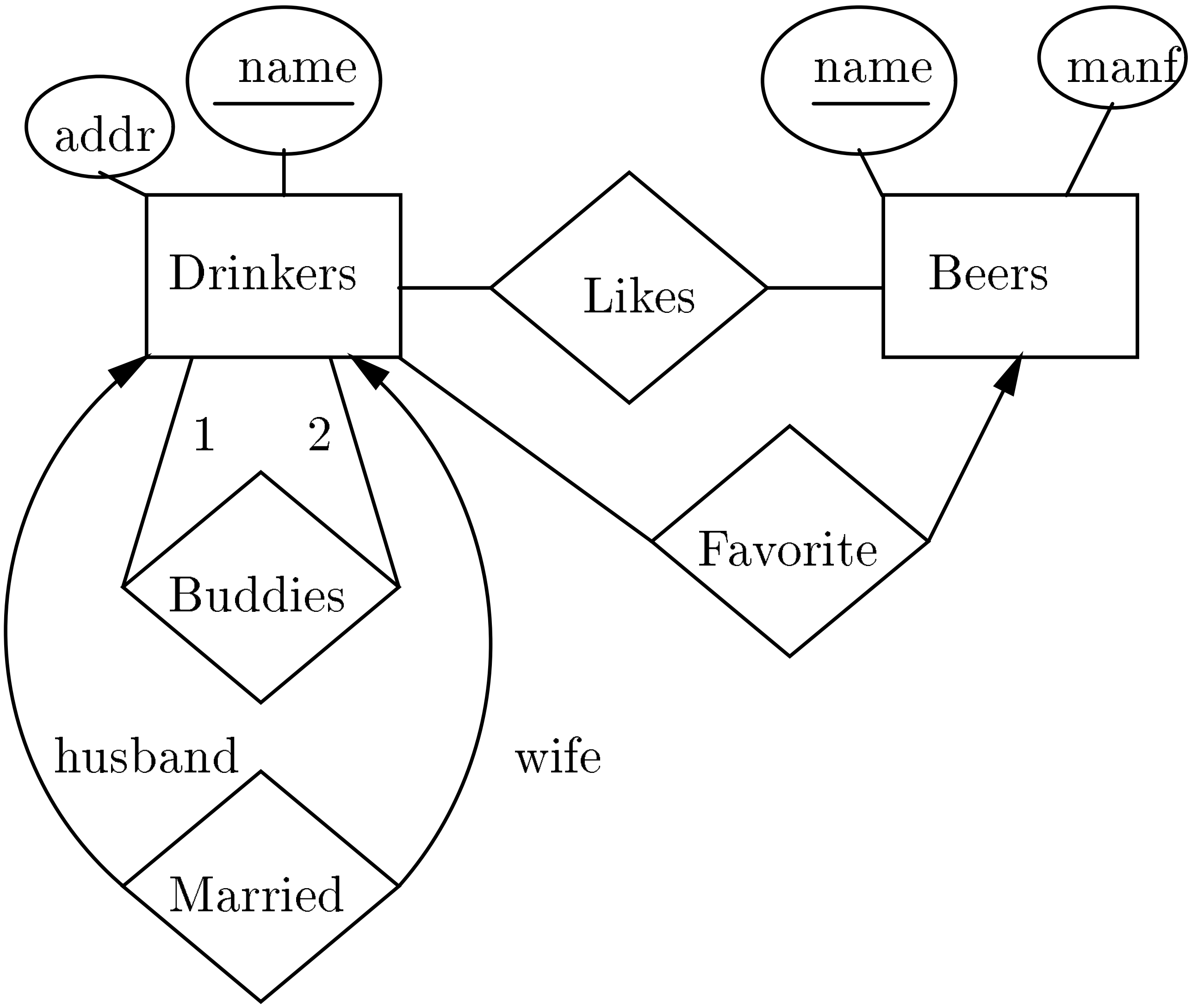
Becomes:

```
Beers(name, manf)
```

E/R Relationships → Relations

Relation has attribute for *key* attributes of each E.S. that participates in the relationship.

- Add any attributes that belong to the relationship itself.
- Renaming attributes OK.
  - Essential if multiple roles for an E.S.
- In the most general case, key for the relation = all key attributes from all the entity sets.
  - However, the relation's key excludes attributes from the "one" side if relationship is many-one.
  - For a one-one relationship, choose which side provides the key of the relation.



Likes(drinker, beer)
Favorite(drinker, beer)
Married(husband, wife)
Buddies(name1, name2)

- For one-one relation `Married`, we can choose either `husband` or `wife` as key.

Combining Relations

Sometimes it makes sense to combine relations.

- Common case: Relation for an E.S. E plus the relation for some many-one relationship from E to another E.S.

Example

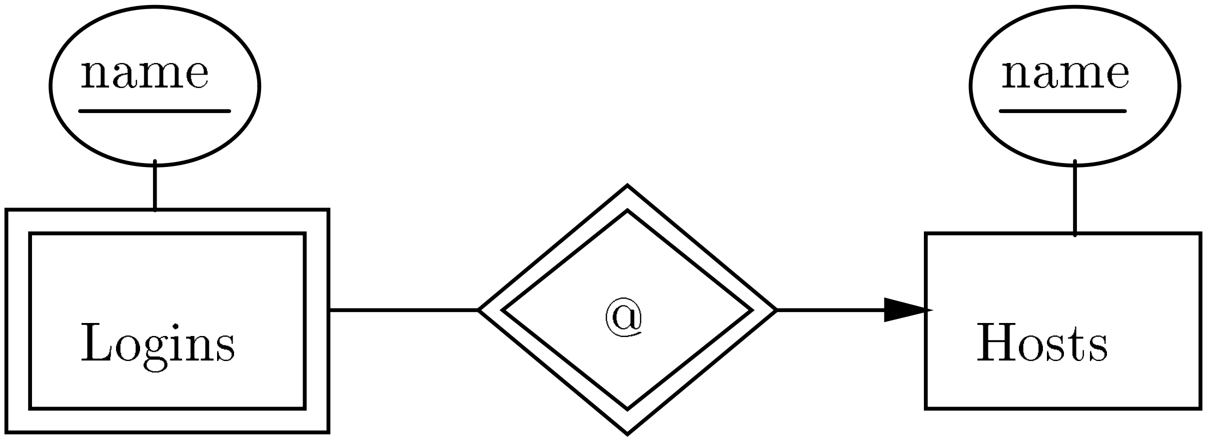
Combine `Drinker(name, addr)` with `Favorite(drinker, beer)` to get `Drinker(name, addr, favBeer)`.

- Danger in pushing this idea too far: redundancy.
- e.g., combining `Drinker` with `Likes` causes the drinker's address to be repeated with every beer he/she likes.
  - ❖ Notice the difference: `Favorite` is many-one; `Likes` is many-many.

Weak Entity Sets, Relationships → Relations

- Relation for a weak E.S. must include its full key (i.e., attributes of related entity sets) as well as its own attributes.
- A many-one relationship that supports a weak entity set (i.e., a double-diamond relationship) yields a relation that is actually redundant and should be deleted from the database schema.

Example



Hosts(<u>hostName</u>)
Logins(<u>loginName</u>, <u>hostName</u>)
At(<u>loginName</u>, <u>hostName</u>, hostName2)

- In `At`, `hostName` and `hostName2` must be the same host, so delete one of them.
- Then, `Logins` and `At` become the same relation; delete one of them.
- In this case, `Hosts`' schema is a subset of `Logins`' schema. Delete `Hosts`?

Same Schema $\neq$ Same Data

There is a subtle assumption in the design process for weak E.S.: the same schema means the same data.

- Not always true. Example:

  Taking(<u>student</u>, <u>course</u>)
  Took(<u>student</u>, <u>course</u>)

- However, when designing a relational schema from E/R, we have only the intent of the E/R diagram to guide us.
- If `Taking` and `Took` were intended to be different concepts, we should make sure both appear in the the diagram.
- Since relations like `Logins` and `At` come from the same E/R feature, we *may* safely assume the relations are the same not only in schema, but in the intended data.